Low Level Design (LLD)

Credit Card Default Prediction

21/03/2024

Devarshi Choudhury

Contents

Abstract

In an era marked by extraordinary advancements in the financial sector, commercial banks are encountering a formidable challenge: the prediction of credit risk amidst evolving financial landscapes. A pivotal concern among these institutions is the ability to foresee the likelihood of credit default among their clients. This project delves into the realm of predictive analytics, aiming to forecast the probability of credit default based on a comprehensive analysis of credit card owners' characteristics and their payment histories.

Financial threats loom large, painting a narrative of the shifting dynamics within commercial banking. The remarkable progress in the financial industry has brought to light an ever-growing need for accurate risk assessment tools. Among these tools, the ability to anticipate credit defaults stands as a critical frontier for banks aiming to navigate the complexities of modern finance.

Imagine the scenario where a seemingly manageable debt spirals out of control due to unforeseen circumstances such as job loss, medical emergencies, or business downturns. It's a narrative many of us can relate to—the missed credit card payments due to oversight or temporary cash flow challenges. However, what happens when these missed payments persist, stretching into months of financial strain?

This project seeks to address precisely this question by harnessing the power of machine learning algorithms. By analysing a diverse range of data points including demographic information like gender, age, marital status, and behavioural patterns such as previous payments and transaction history, we aim to develop a robust model. This model will not only predict the likelihood of a customer becoming a defaulter but also empower banks with actionable insights to mitigate risks effectively.

In a world where financial stability is paramount, this project endeavours to provide commercial banks with a predictive tool that can potentially redefine their risk management strategies. Through the lens of machine learning, we embark on a journey to enhance the industry's ability to foresee and navigate the intricate landscape of credit default risks.

1. Introduction

1.1 What is a Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for the Credit Card Default prediction code. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 Scope

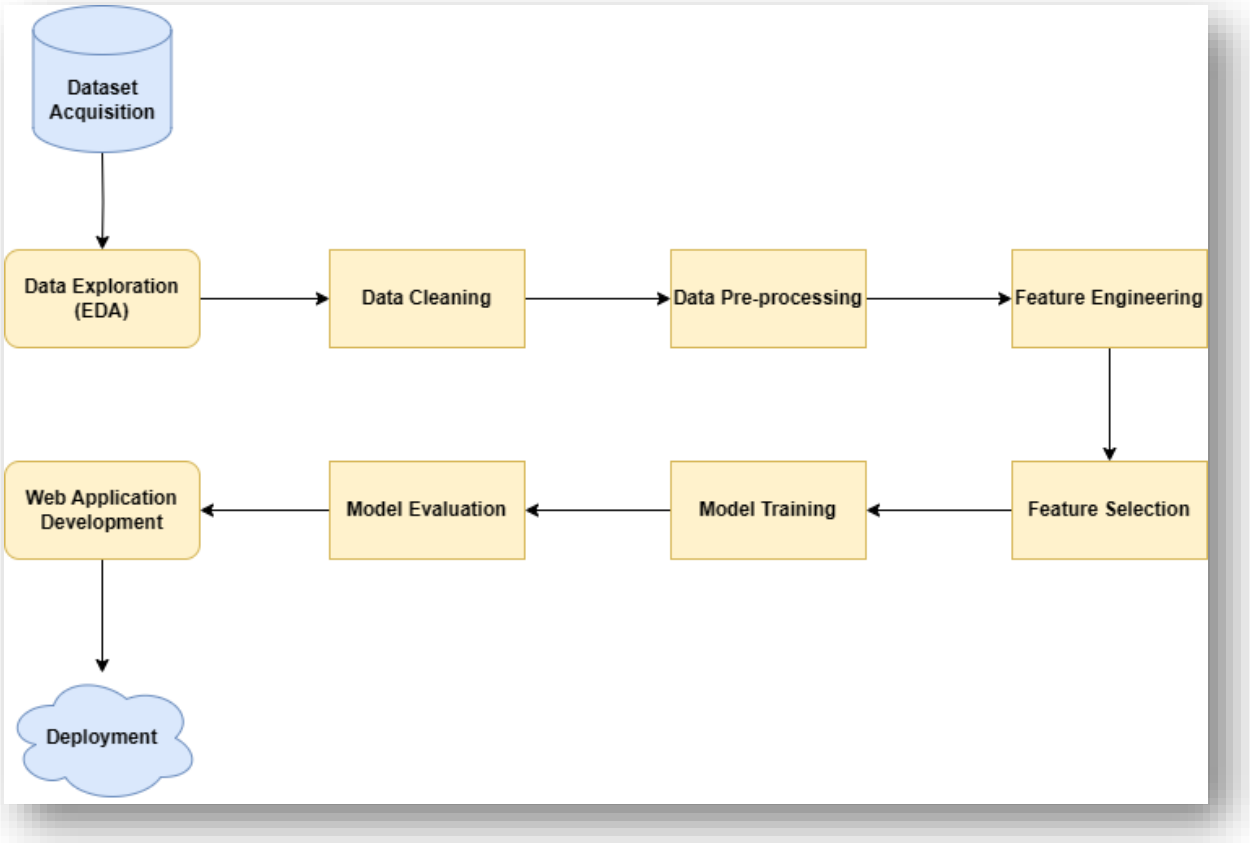
Low-level design (LLD) is a component-level design process that follows a step-by step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

1.1 Definitions

Term	Description
IDE	Integrated Development Environment
EDA	Exploratory Data Analysis
VS Code	Visual Studio Code
LightGBM	Light Gradient-Boosting Machine

2. System Architecture

a. **Frontend:** Streamlit for creating the user interface
b. **Backend:** Python for data processing, model training, and prediction.



3. Dataset Information:

ID: ID of each client

LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary = credit)

SEX: Gender (1=male, 2=female)

EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)

MARRIAGE: Marital status (1=married, 2=single, 3=others)

AGE: Age in years

PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)

PAY_2: Repayment status in August, 2005 (scale same as above)

PAY_3: Repayment status in July, 2005 (scale same as above)

PAY_4: Repayment status in June, 2005 (scale same as above)

PAY_5: Repayment status in May, 2005 (scale same as above)

PAY_6: Repayment status in April, 2005 (scale same as above)

BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)

BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)

BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)

BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)

BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)

BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)

PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)

PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)

PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)

PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)

PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)

PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)

default.payment.next.month: Default payment (1=yes, 0=no)

4. Architecture Description

4.1 Data Description:

The dataset used in this project is the "Default of Credit Card Clients Dataset" obtained from Kaggle:

- Data Source: Default of Credit Card Clients Dataset
- This dataset contains 25 attributes with 30,000 observations for both training and testing sets.
- Key features include credit limit, gender, education level, marital status, age, payment status for multiple months, and bill amounts.

4.2 Data Transformation:

- The original CSV dataset is converted into a pandas DataFrame.
- After reading the dataset, two DataFrames train and test are concatenated into one DataFrame for further processing.

4.3 Exploratory Data Analysis (EDA):

- EDA is conducted to investigate the dataset, identify patterns, anomalies (outliers), and form hypotheses.
- It involves statistical analysis, visualizations, and summary statistics to gain insights into the data.

4.4 Data Insertion into Database:

- Creation of a database and connection establishment.
- Table creation within the database to store the dataset.
- Insertion of dataset into the database for storage and retrieval.

4.5 Export Data from Database:

- The stored dataset is exported from the database as a CSV file.
- This exported CSV file is used for further data preprocessing and model training.

4.6 Data Cleaning:

- Data cleansing involves detecting and correcting (or removing) corrupt or inaccurate records.
- Tasks include handling missing values, correcting data types, and removing duplicates.

4.7 Data Pre-processing:

- Pre-processing prepares the raw data for machine learning models.
- Steps include scaling numerical features, encoding categorical variables, and handling imbalanced data.

4.8 Feature Engineering:

- Feature engineering is utilized to create new features that enhance model performance.
- In this project, features like 'Credit Utilization Ratio', 'Age Groups', and 'Payment Status Trends' are engineered from existing attributes.

4.9 Feature Selection:

- Feature selection is crucial to improve model efficiency and reduce overfitting.
- Techniques like Variance Inflation Factor (VIF) are employed to detect and potentially remove highly correlated features.

4.10 Model Building:

- Machine learning models are built using various algorithms such as Random Forest, LightGBM, and XGBoost.
- Training the models involves splitting the data into training and testing sets, fitting the model, and evaluating its performance.

4.11 Random Forest Model:

- Random Forest is a meta-estimator that fits multiple decision tree classifiers on random sub-samples of the dataset.
- It improves predictive accuracy by averaging results and helps control overfitting.

4.12 LightGBM Model:

- LightGBM is a gradient boosting framework that uses tree-based learning algorithms.
- It is known for its speed and efficiency, particularly with large datasets.
- LightGBM builds trees vertically, leaf-wise rather than level-wise, reducing loss more efficiently.

4.13 XGBoost Model:

- XGBoost is an optimized distributed gradient boosting library.
- It is highly efficient, scalable, and flexible, making it a popular choice for machine learning competitions.
- XGBoost uses a gradient boosting framework and is known for its speed and performance.

These models (Random Forest, LightGBM, XGBoost) were trained and evaluated to predict credit card defaults in the project. Each model has its strengths and was tested to determine the best performing one for the given dataset.

4.14 Web Application with Streamlit:

- Streamlit is used to create the web application interface.
- The app includes input widgets for user input, prediction functionality, and displays the prediction results.
- Users can input features like credit limit, gender, education, marital status, age, payment status, and bill amounts.
- Upon entering the input values, the application predicts in real-time whether the customer is likely to default.

4.15 Deployment:

- The web application can be deployed using platforms like Heroku, AWS, or Streamlit Sharing.
- Deployment involves setting up a web server to host the app and making it accessible via a URL.

4.16 Technologies Used:

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn are used to build such a model

- Google Colab was used as the work environment
- For Visualization of the plots, Matplotlib and Seaborn are used.
- VS Code was used to build app.py file for web application.
- GitHub is used as Version Management System.
- Streamlit for cloud deployment.
- Statsmodels library is used for checking relationship with target variable.

5. Unit Test Cases:

Test Case Description	Pre-requisite	Expected Result
Verify user can view input data in the application	Application is accessible in the browser	Users can see their input data
Verify user can edit their information in the form	Application is responsive	User should be able to edit all input fields
Verify user can submit input values with Submit button	Application is accessible	User should be able to submit values
Verify wrong information is not submitted by the user	Application is secure and responsive	Correct information should be submitted